Amendments to the Drawings

The attached sheet replaces the original sheet including Fig. 1. In Fig. 1, the destinations/sources of input/output lines of elements 125 and 125 are added.

Attachment: Replacement Sheet

REMARKS

At the time of the Office Action dated October 3, 2005, claims 1-20 were pending. In this Amendment, claims 1, 3, 4, 9-20 have been amended, claim 19 has been canceled, without prejudice, and the specification and drawings have also been amended. Care has been exercised to avoid the introduction of new matter. Adequate descriptive support for the present Amendment should be apparent throughout the originally filed disclosure as, for example, the depicted embodiments and related discussion thereof in the written description of the specification.

Oath/Declaration

The Examiner asserted that the oath or declaration is defective because it does not contain signatures for all inventors. In response, Applicants note that the six sets of the declarations were filed on October 3, 2003. The six sets of the declarations as a whole contains the signatures for all the inventors. Applicants respectfully request the Examiner to review the declarations, and clarify the record by acknowledging that the declarations are not defective. Attached are copies of the declaration and the stamped post card for the Examiner's review.

Drawings

The Examiner has objected to Fig. 1 because the destinations/sources of input/output lines attached to element 125 and 126 are not identified. In response, Fig. 1 has been amended to clarify the destinations/sources of input/output lines of element 125 and 126. Adequate descriptive support for this amendment can be found on, for example, page 16, line 23 to page 17, line 7 of the specification. Withdrawal of the objection to the drawings is respectfully solicited.

Specification

The Examiner also objected to the specification. In response, Applicants has submitted a Substitute Specification including changes addressing the Examiner's objections (see paragraph 5, subparagraphs a, c, d and e of the Office Action). The abstract has been amended to eliminate the expressions "'unit' by 'unit'" and "constituted by" (see subparagraph b).

In paragraph 5, subparagraph f of the Office Action, the Examiner requested the document "UDF specification 2.2.6.4." and "UDF specification 2.2.6" to be submitted. In response, Applicants attach the document regarding "UDF specification 2.2.6.4." and "UDF specification 2.2.6" to this Amendment for the Examiner's review.

Applicants, therefore, respectfully solicit withdrawal of the objection to the specification.

Claims 1-20 have been rejected under 35 U.S.C. §112, second paragraph.

The Examiner asserted that the claims are indefinite for failing to particularly point out and distinctly claim the subject matter which applicants regard as the invention, and are incomplete for omitting essential elements and/or structural cooperative relationship of elements (see paragraphs 8 and 9 of the Office Action).

In response, Applicants have amended claims 1, 3, 4, 9-20 based on the Examiner's comments, thereby overcoming the stated bases for the rejection of the claims. Applicants, therefore, respectfully solicit withdrawal of the rejection of claims 1-20 under 35 U.S.C. §112, second paragraph, and favorable consideration thereof.

Claim 19 has been rejected under 35 U.S.C. §101 and under 35 U.S.C. §112, first paragraph.

It is noted that the rejection of claim 19 has been rendered moot by cancellation of the claim. Withdrawal of the rejection of claim 19 under 35 U.S.C. §§102 and 112, first paragraph is respectfully solicited.

Conclusion

It should, therefore, be apparent that the imposed rejections have been overcome and that all pending claims are in condition for immediate allowance. Favorable consideration is, therefore, respectfully solicited.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account 500417 and please credit any excess fees to such deposit account.

Respectfully submitted,

McDERMOTT WILL & EMERY LLP

Tomoki Tanida
Limited Recognition No. L0098

600 13th Street, N.W.
Washington, DC 20005-3096
Phone: 202.756.8000 AJS:TT
Facsimile: 202.756.8087
Date: January 3, 2006

Please recognize our Customer No. 20277 as our correspondence address.

Note: Time zones West of Coordinated Universal Time have negative offsets. For example, Eastern Standard Time is -300 minutes; Eastern Daylight Time is -240 minutes.

Note: Implementations on systems that support time zones should interpret unspecified time zones as Coordinated Universal Time. Although not a requirement, this interpretation has the advantage that files generated on systems that do not support time zones will always appear to have the same time stamps on systems that do support time zones, irrespective of the interpreting system's local time zone.

2.1.5 Entity Identifier

```
struct EntityID {          /* ECMA 167 1/7.4 */
        Uint8              Flags;
        char               Identifier[23];
        char               IdentifierSuffix[8];
}
```

UDF classifies *Entity Identifiers* into 4 separate types as follows:

- *Domain Entity Identifiers*
- *UDF Entity Identifiers*
- *Implementation Entity Identifiers*
- *Application Entity Identifiers*

The following sections describe the format and use of *Entity Identifiers* based upon the different types mentioned above.

2.1.5.1 Uint8 Flags

Self-explanatory.

Shall be set to ZERO.

2.1.5.2 char Identifier

Unless stated otherwise in this document this field shall be set to an identifier that uniquely identifies the implementation. This methodology will allow for identification of the implementation responsible for creating structures recorded on media interchanged between different implementations.

If an implementation updates existing structures on the media written by other implementations the updating implementation shall set the *Identifier* field to a value that uniquely identifies the updating implementation.

The following table summarizes the *Entity Identifier* fields defined in the ECMA 167 standard and this document and shows to what values they shall be set.

Entity Identifiers

Descriptor	Field	ID Value	Suffix Type
Primary Volume Descriptor	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
Primary Volume Descriptor	Application ID	*"*Application ID"*	Application Identifier Suffix
Implementation Use Volume Descriptor	Implementation Identifier	"*UDF LV Info"	UDF Identifier Suffix
Implementation Use Volume Descriptor	Implementation ID (in Implementation Use field)	*"*Developer ID"*	Implementation Identifier Suffix
Partition Descriptor	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
Partition Descriptor	Partition Contents	*"+NSR03"*	Application Identifier Suffix
Logical Volume Descriptor	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
Logical Volume Descriptor	Domain ID	"*OSTA UDF Compliant"	DOMAIN Identifier Suffix
File Set Descriptor	Domain ID	"*OSTA UDF Compliant"	DOMAIN Identifier Suffix
File Identifier Descriptor	Implementation Use	*"*Developer ID"*	Implementation Identifier Suffix *(optional)*
File Entry	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
Device Specification Extended Attribute	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
UDF Implementation Use Extended Attribute	Implementation ID	*See 3.3.4.5*	UDF Identifier Suffix
Non-UDF Implementation Use Extended Attribute	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
UDF Application Use Extended Attribute	Application ID	*See 3.3.4.6*	UDF Identifier Suffix
Non-UDF Application Use Extended Attribute	Application ID	*"*Application ID"*	Application Identifier Suffix
UDF Unique ID Mapping Data	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix
Power Calibration Table Stream	Implementation ID	*"*Developer ID"*	Implementation Identifier Suffix

Logical Volume Integrity Descriptor	Implementation ID (in Implementation Use field)	*"*Developer ID"*	Implementation Identifier Suffix
Partition Integrity Entry	Implementation ID	N/A	N/A
Virtual Partition Map	Partition Type Identifier	""*UDF Virtual Partition"	UDF Identifier Suffix
Virtual Allocation Table	Implementation Use	*"*Developer ID"*	Implementation Identifier Suffix *(optional)*
Sparable Partition Map	Partition Type Identifier	""*UDF Sparable Partition"	UDF Identifier Suffix
Sparing Table	Sparing Identifier	""*UDF Sparing Table"	UDF Identifier Suffix

NOTE: The value of the Entity Identifier field is interpreted as a sequence of bytes, and not as a dstring specified in CS0. For ease of use the values used by UDF for this field are specified in terms of ASCII character strings. The actual sequence of bytes used for the Entity Identifiers defined by UDF are specified in section 6.2.

NOTE: In the *ID Value* column in the above table *"*Application ID"* refers to an identifier that uniquely identifies the writer's application.

In the *ID Value* column in the above table *"*Developer ID"* refers to an Entity Identifier that uniquely identifies the current implementation. The value specified should be used when a new descriptor is created. Also, the value specified should be used for an existing descriptor when anything within the scope of the specified EntityID field is modified.

NOTE: The value chosen for a *"*Developer ID"* should contain enough information to identify the company and product name for an implementation. For example, a company called *XYZ* with a UDF product called *DataOne* might choose *"*XYZ DataOne"* as their developer ID. Also in the suffix of their developer ID they may choose to record the current version number of their *DataOne* product. This information is extremely helpful when trying to determine which implementation wrote a bad structure on a piece of media when multiple products from different companies have been recording on the media.

The *Suffix Type* column in the above table defines the format of the suffix to be used with the corresponding Entity Identifier. These different suffix types are defined in the following paragraphs.

NOTE: All *Identifiers* defined in this document (appendix 6.1) shall be registered by OSTA as UDF *Identifiers*.

2.2.4.7 byte PartitionMaps

For the purpose of interchange partition maps shall be limited to Partition Map type 1, except type 2 maps as described in this document (2.2.8 and 2.2.9).

2.2.5 Unallocated Space Descriptor

```
struct UnallocatedSpaceDesc {                    /* ECMA 167 3/10.8 */
        struct tag      DescriptorTag;
        Uint32          VolumeDescriptorSequenceNumber;
        Uint32          NumberofAllocationDescriptors;
        extent_ad.      AllocationDescriptors[];
}
```

This descriptor shall be recorded, even if there is no free volume space. The first 32768 bytes of the Volume space shall not be used for the recording of ECMA 167 structures. This area shall not be referenced by the Unallocated Space Descriptor or any other ECMA 167 descriptor.

2.2.6 Logical Volume Integrity Descriptor

```
struct LogicalVolumeIntegrityDesc {              /* ECMA 167 3/10.10 */
        struct tag      DescriptorTag,
        Timestamp       RecordingDateAndTime,
        Uint32          IntegrityType,
        struct extend_ad NextIntegrityExtent,
        byte            LogicalVolumeContentsUse[32],
        Uint32          NumberOfPartitions,
        Uint32          LengthOfImplementationUse,
        Uint32          FreeSpaceTable [],
        Uint32          SizeTable[],
        byte            ImplementationUse[]
}
```

The *Logical Volume Integrity Descriptor* is a structure that shall be written any time the contents of the associated Logical Volume is modified. Through the contents of the *Logical Volume Integrity Descriptor* an implementation can easily answer the following useful questions:

1) Are the contents of the Logical Volume in a consistent state?

2) When was the last date and time that anything within the Logical Volume was modified?

3) What is the total Logical Volume free space in logical blocks?

4) What is the total size of the Logical Volume in logical blocks?

5) What is the next available UniqueID for use within the Logical Volume?

6) Has some *other* implementation modified the contents of the logical volume since the last time that the *original* implementation, which created the logical volume, accessed it.

2.2.6.1 byte LogicalVolumeContentsUse

See section 3.2.1 for information on the contents of this field.

2.2.6.2 Uint32 FreeSpaceTable

Since most operating systems require that an implementation provide the true free space of a Logical Volume at mount time it is important that these values be maintained for all non-virtual partitions. The optional value of #FFFFFFFF, which indicates that the amount of available free space is not known, shall not be used for non-virtual partitions. For virtual partitions the FreeSpaceTable shall be set to #FFFFFFFF.

NOTE: The FreeSpaceTable is guaranteed to be correct only when the *Logical Volume Integrity Descriptor* is closed.

2.2.6.3 Uint32 SizeTable

Since most operating systems require that an implementation provide the total size of a Logical Volume at mount time it is important that these values be maintained for all non-virtual partitions. The optional value of #FFFFFFFF, which indicates that the partition size is not known, shall not be used for non-virtual partitions. For virtual partitions the SizeTable shall be set to #FFFFFFFF.

2.2.6.4 byte ImplementationUse

The *ImplementationUse* area for the *Logical Volume Integrity Descriptor* shall be structured as follows:

ImplementationUse format

RBP	Length	Name	Contents
0	32	ImplementationID	EntityID
32	4	Number of Files	Uint32
36	4	Number of Directories	Uint32
40	2	Minimum UDF Read Revision	Uint16
42	2	Minimum UDF Write Revision	Uint16
44	2	Maximum UDF Write Revision	Uint16
46	??	Implementation Use	byte

Implementation ID - The implementation identifier *EntityID* of the implementation which last modified anything within the scope of this *EntityID*. The scope of this *EntityID* is the Logical Volume Descriptor, and the contents of the associated Logical Volume. This field allows an implementation to identify which implementation last modified the contents of a Logical Volume.

Number of Files - The current number of files in the associated Logical Volume. This information is needed by the Macintosh OS. All implementations shall maintain this information. NOTE: This value does not include Extended Attributes or streams as part of the file count.

Number of Directories - The current number of directories in the associated Logical Volume. This information is needed by the Macintosh OS. All implementations shall maintain this information.
NOTE: The root directory shall be included in the directory count. The directory count does not include stream directories.

Minimum UDF Read Revision - Shall indicate the minimum recommended revision of the UDF specification that an implementation is required to support to successfully be able to read all potential structures on the media. This number shall be stored in binary coded decimal format, for example #0150 would indicate revision 1.50 of the UDF specification.

Minimum UDF Write Revision - Shall indicate the minimum revision of the UDF specification that an implementation is required to support to successfully be able to modify all structures on the media. This number shall be stored in binary coded decimal format, for example #0150 would indicate revision 1.50 of the UDF specification.

Maximum UDF Write Revision - Shall indicate the maximum revision of the UDF specification that an implementation that has modified the media has supported. An implementation shall update this field only if it has modified the media and the level of the UDF specification it supports is higher than the current value of this field. This number shall be stored in binary coded decimal format, for example #0150 would indicate revision 1.50 of the UDF specification.

Implementation Use - Contains implementation specific information unique to the implementation identified by the Implementation ID.